JAVA.UTIL.ARRAYDEQUE CLASS

 Copyright © tutorialspoint.com

Introduction

The **java.util.ArrayDeque** class provides resizable-array and implements the **Deque** interface. Following are the important points about Array Deques:

- Array deques have no capacity restrictions so they grow as necessary to support usage.
- They are not thread-safe; in the absence of external synchronization.
- They do not support concurrent access by multiple threads.
- Null elements are prohibited in the array deques.
- They are faster than Stack and LinkedList.

This class and its iterator implement all of the optional methods of the **Collection** and **Iterator** interfaces.

Class declaration

Following is the declaration for **java.util.ArrayDeque** class:

```
public class ArrayDeque<E>
   extends AbstractCollection<E>
      implements Deque<E>, Cloneable, Serializable
```

Here **<E>** represents an Element, which could be any class. For example, if you're building an array list of Integers then you'd initialize it as

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

Class constructors

S.N.	Constructor & Description
1	**ArrayDeque()** This constructor is used to create an empty array deque with an initial capacity sufficient to hold 16 elements.
2	**ArrayDeque(Collection<? extends E> c)** This constructor is used to create a deque containing the elements of the specified collection.
3	**ArrayDeque(int numElements)** This constructor is used to create an empty array deque with an initial capacity sufficient to hold the specified number of elements.

Class methods

S.N.	Method & Description

1	boolean add(E e) This method inserts the specified element at the end of this deque.
2	void addFirst(E e) This method inserts the specified element at the front of this deque.
3	void addLast(E e) This method inserts the specified element at the end of this deque.
4	void clear() This method removes all of the elements from this deque.
5	ArrayDeque<E> clone() This method returns a copy of this deque.
6	boolean contains(Object o) This method returns true if this deque contains the specified element.
7	Iterator<E> descendingIterator() This method returns an iterator over the elements in this deque in reverse sequential order.
8	E element() This method retrieves, but does not remove, the head of the queue represented by this deque.
9	E getFirst() This method retrieves, but does not remove, the first element of this deque.
10	E getLast() This method retrieves, but does not remove, the last element of this deque.
11	boolean isEmpty() This method returns true if this deque contains no elements.
12	Iterator<E> iterator() This method returns an iterator over the elements in this deque.
13	boolean offer(E e) This method inserts the specified element at the end of this deque.
14	boolean offerFirst(E e) This method inserts the specified element at the front of this deque.
15	boolean offerLast(E e) This method inserts the specified element at the end of this deque.
16	E peek() This method retrieves, but does not remove, the head of the queue represented by this deque, or returns null if this deque is empty.
17	E peekFirst() This method retrieves, but does not remove, the first element of this deque, or returns null if this deque is empty.
18	E peekLast() This method retrieves, but does not remove, the last element of this deque, or returns null if this deque is empty.

19	E poll() This method retrieves and removes the head of the queue represented by this deque, or returns null if this deque is empty.
20	E pollFirst() This method retrieves and removes the first element of this deque, or returns null if this deque is empty.
21	E pollLast() This method retrieves and removes the last element of this deque, or returns null if this deque is empty.
22	E pop() This method pops an element from the stack represented by this deque.
23	void push(E e) This method pushes an element onto the stack represented by this deque.
24	E remove() This method retrieves and removes the head of the queue represented by this deque.
25	boolean remove(Object o) This method removes a single instance of the specified element from this deque.
26	E removeFirst() This method retrieves and removes the first element of this deque.
27	boolean removeFirstOccurrence(Object o) This method removes the first occurrence of the specified element in this deque.
28	E removeLast() This method retrieves and removes the last element of this deque.
29	boolean removeLastOccurrence(Object o) This method removes the last occurrence of the specified element in this deque.
30	int size() This method returns the number of elements in this deque.
31	object[] toArray() This method returns an array containing all of the elements in this deque in proper sequence.